Amendments to the Claims:

1. (Previously Presented) A processor that predicts aliasing between read type instructions and write type instructions based at least in part on respective displacements between the read type and write type instructions and on previous detection of respective aliasings between the read type instructions and the write type instructions, and that bypasses data from the write type instructions to the corresponding predicted to alias read type instructions using register information of the aliasing write type instructions.

2. (Previously Presented) The processor of claim 1 wherein the data is bypassed if a threshold number of repeated aliases are detected.

3. (Original) The processor of claim 1 that includes encodings of read type instruction information, write type instruction information, and repeat aliasing.

4. (Original) The processor of claim 3 wherein the encodings comprise:

a read type instruction aliasing predictor table that indicates a static instruction identifier for a read type instruction, a displacement between the indicated read type instruction and a previously aliased write type instruction, and an alias prediction confidence indicator that indicates confidence of alias predictions;

a write type instruction aliasing predictor table that indicates the static instruction identifier for a write type instruction; and

an aliasing predictor management table that indicates a rename register identifier, an alias prediction counter that indicates a number of alias predictions, and a dynamic instruction identifier.

5. (Original) The processor of claim 4 wherein the static instruction identifier includes an address for the read or write-type instruction.

6. (Original) The processor of claim 4 wherein the dynamic instruction identifier monotonically increases with execution of a program that includes the instructions.

7. (Original) The processor of claim 4 that reduces the alias confidence prediction indicator for a read type instruction indicated in the read type instruction aliasing predictor table

if the aliasing predictor management table does not indicate a write type instruction that corresponds to the read type instruction and the read type instruction's corresponding displacement.

8. (Original) The processor of claim 4 that reduces the alias confidence prediction indicator for a read type instruction indicated in the read type instruction aliasing predictor table if a misprediction of the read type instruction occurs.

9. (Original) The processor of claim 4 wherein the read type instruction aliasing predictor table includes a validity flag that indicates whether a threshold number of aliasings have been detected.

10. (Original) The processor of claim 1 wherein data bypasses comprise the processor substituting a register move instruction for the read type instruction.

11. (Original) The processor of claim 10, wherein a loadCheck instruction is inserted, which when executed by the processor, causes the processor to verify the predicted aliasing.

12. (Original) The processor of claim 10 wherein the register move instruction includes an integer-to-integer move instruction, a floating point-to-floating point move instruction, an integer-to-floating point move instruction, and a floating point-to-integer move instruction.

13. (Original) The processor of claim 1 wherein data bypasses comprise the processor mapping the read type instruction's destination register to the write type instruction's source register.

14. (Original) The processor of claim 13 that replaces the read type instruction with a loadCheck instruction, which when executed by the processor, causes the processor to verify the predicted aliasing.

15. (Previously Presented) The processor of claim 14 wherein the processor's verification of the predicted aliasing comprises interrogation of a data hazard detection module

to ascertain whether addresses of the predicted to alias write type instruction and the read type instruction match, and verification of absence of intervening matching write type instructions.

16. (Currently Amended) A method comprising:

in a register rename stage, tracking a write type instruction and a read type instruction, instances of which have previously been indicated as aliased; [[and]]

predicting a current instance of the read type instruction will alias with a current instance of the write type instruction [[if]] when displacement between the current instance of the read type instruction and the current instance of the write type instruction matches displacement between previous aliased instances of the read type instruction and write type instruction; and

bypassing data of the write type instruction to the read type instruction with register information of the write type instruction.

17. (Previously Presented) The method of claim 16 wherein the displacement is measured with dynamic instruction identifiers, wherein the dynamic instruction identifiers identify corresponding instances of instructions with respect to program execution.

18. (Original) The method of claim 16 wherein the write type instruction and the read type instruction are tracked with their static identifier, wherein the static identifier identifies an instruction in a program and remains static during program execution.

19. (Original) The method of claim 18 wherein the static identifier includes an instruction address.

20. (Original) The method of claim 16 wherein a read type instruction includes a load instruction, a load halfword instruction, a load byte instruction, a load float instruction, a load double instruction, and a load multiple instruction.

21. (Original) The method of claim 16 wherein the write type instruction includes a store instruction, a store byte instruction, a store float instruction, a store double instruction, a store multiple instruction, and a store halfword instruction.

22. (Canceled)

23. (Currently Amended) The method of claim [[22]]16 wherein bypassing comprises mapping the read type instruction's destination register to the write type instruction's source register.

24. (Original) The method of claim 23 further comprising replacing the read type instruction with a loadCheck instruction, wherein the loadCheck instruction causes interrogation of a data hazard detection module to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and to ascertain whether there are any intervening matching write type instructions.

25. (Currently Amended) The method of claim [[22]]16 wherein bypassing comprises converting the read type instruction to a register move instruction.

26. (Original) The method of claim 25 further comprising inserting a loadCheck instruction, wherein the loadCheck instruction causes interrogation of a data hazard detection logic to ascertain whether addresses of the write type instruction and the read type instruction predicted to alias with the write type instruction match, and to ascertain whether there are any intervening matching write type instructions.

27. (Previously Presented) The method of claim 16 embodied as a computer program product encoded in one or more machine-readable storage media.

28. (Currently Amended) A method comprising:

observing repeated aliasing between instances of a write type instruction and a read type instruction based at least in part on static identifiers of the instructions;

determining a displacement between the aliasing instances of the write type instruction and the read type instruction based on dynamic identifiers of the instruction instances;

predicting aliasing between a current instance of the read type instruction as identified by the static identifier thereof and a subsequent instance of the write type instruction identified with a dynamic identifier determined with a dynamic identifier of the current instance of the read type instruction and the displacement; and

bypassing data of the subsequent instance of the write type instruction to the current instance of the read type instruction with register information of the subsequent instance of the write type instruction.

29. (Canceled)

30. (Currently Amended) The method of claim [[29]]28 wherein bypassing the data comprises mapping the data destination of the current instance of the read type instruction to the data source of the subsequent instance of the write type instruction.

31. (Previously Presented) The method of claim 30 further comprising substituting a loadCheck instruction for the current instance of the read type instruction, wherein execution of the loadCheck instruction causes interrogation of a data hazard detection module to ascertain whether addresses of the current instance of the read type instruction and the subsequent instance of the write type instruction match.

32. (Original) The method of claim 31 wherein execution of the loadCheck instruction further causes verifying the absence of intervening matching write type instructions.

33. (Currently Amended) The method of claim [[29]]28 wherein bypassing the data comprises substituting a register move instruction for the read type instruction, wherein the move instruction moves data from the data source of the write type instruction to the data destination of the read type instruction.

34. (Original) The method of claim 28 wherein the dynamic identifiers monotonically increase with execution of a program that includes the instructions.

35. (Original) The method of claim 28 wherein the static identifiers include instruction addresses.

36. (Original) The method of claim 28 wherein the aliasing is predicted if the number of observed repeat aliasings exceeds a threshold.

37. (Previously Presented) The method of claim 28 embodied as a computer program product encoded in one or more machine-readable storage media.

38. (Previously Presented) A method comprising:

detecting aliasing between a first instance of a read type instruction and a first instance of a write type instruction;

determining displacement between the first instance of the read type instruction and the first instance of the write type instruction, wherein the displacement is with respect to program execution;

observing repeated aliasing between subsequent instances of the read type instruction and the write type instruction;

selecting a current instance of the write type instruction based at least in part on the displacement and a current instance of the read type instruction; and

bypassing data from a data source of the current instance of the write type instruction to a data destination of the current instance of the read type instruction.

39. (Previously Presented) The method of claim 38 further comprising verifying that the current instance of the write type instruction aliases with the current instance of the read type instruction.

40. (Previously Presented) The method of claim 38 wherein data bypass comprises changing the current instance of the read type instruction to a register move instruction.

41. (Previously Presented) The method of claim 40 further comprising inserting a loadCheck instruction, wherein the loadCheck instruction causes verification that the current instance of the read type instruction and the current instance of the write type instruction alias and verification of the absence of one or more intervening write type instructions.

42. (Previously Presented) The method of claim 38 wherein data bypasses comprise mapping the current instance of the read type instruction's architectural destination register to the current instance of the write type instruction's rename source register.

43. (Previously Presented) The method of claim 42 further comprising changing the current instance of the read type instruction to a loadCheck instruction, wherein the loadCheck instruction causes verification that the current instances of the read type instruction and the write type instruction alias and verification of the absence of one or more intervening write type instructions.

44. (Previously Presented) The method of claim 38 wherein the instances of the write type instruction have a same static identifier and different dynamic identifiers.

45. (Original) The method of claim 44 wherein the static identifiers include instruction addresses.

46. (Previously Presented) The method of claim 38 wherein the displacement is based at least in part on the dynamic identifiers of the instruction instances, wherein the dynamic identifiers monotonically increase with execution of a program that includes the instructions.

47. (Previously Presented) The method of claim 38 embodied as a computer program product encoded in one or more machine-readable storage media.

48. (Previously Presented) A computer program product encoded in one or more machine-readable storage media, the computer program product comprising:

a first sequence of instructions executable to, update a first encoding with a read type instruction's static identifier and a displacement between an instance of the read type instruction and an instance of a write type instruction observed as aliasing with the read type instruction instance if the read type instruction's static identifier is not indicated in the first encoding and to update the first encoding to indicate repeat aliasing if the read type instruction's static identifier is already indicated in the first encoding, update a second encoding with the write type instruction's static identifier;

a second sequence of instructions executable to update a third encoding with a dynamic identifier of an instance of a write type instruction if the static identifier thereof is indicated in the second encoding; and

a third sequence of instructions executable to bypass data from an instance of a write type instruction to an instance of a read type instruction with register information of the write type instruction instance based at least in part on displacement between the instances as indicated by corresponding dynamic identifiers.

49. (Previously Presented) The computer program product of claim 48 wherein data bypassing comprises the third sequence of instructions executable to map the read type instruction's destination register to the write type instruction's source register.

50. (Original) The computer program product of claim 49 wherein the read type instruction's destination register includes an architectural register and the write type instruction's source register includes a rename register.

51. (Previously Presented) The computer program product of claim 50 wherein the third sequence of instructions are further executable to re' lace the read type instruction instance with a loadCheck instruction, which when executed causes verification that the read type instruction and the write type instruction alias to the same address and verification of the absence of one or more intervening write type instructions aliasing to the same address.

52. (Previously Presented) The computer program product of claim 48 wherein bypassing data comprises the third sequence of instructions executable to replace the read type instruction with a register move instruction.

53. (Previously Presented) The computer program product of claim 52, further comprising the third sequence of instructions executable to insert a loadCheck instruction proximate with the register move instruction, wherein execution of the loadCheck instruction causes verification that the read type instruction and the write type instruction alias to the same address and verification of the absence of one or more intervening write type instructions aliasing to the same address.

54. (Previously Presented) An apparatus comprising:

a data hazard detection module; and

means for predicting aliasing between a current instance of a read type instruction and a current instance of a write type instruction based on displacement between the current instance of the instructions and displacement between previously observed aliased instances of the read type instruction and a the write type instruction.

55. (Original) The apparatus of claim 54 further comprising means for bypassing data from the write type instruction to the read type instruction with register information of the write type instruction.

56. (Original) The apparatus of claim 54 wherein the read type instruction includes a load instruction, a load halfword instruction, a load byte instruction, a load float instruction, a load double instruction, and a load multiple instruction.

57. (Original) The apparatus of claim 54 wherein the write type instruction includes a store instruction, a store byte instruction, a store float instruction, a store double instruction, a store multiple instruction, and a store halfword instruction.

58. (Previously Presented) An apparatus comprising:

a data hazard detection module; and

rename unit coupled with the data hazard detection module, the rename unit to rename registers of instructions and to predict aliasing between instances of read type instructions and instances of write type instructions based at least in part on respective displacements between the instruction instances, wherein the rename unit includes one or more structures operable to, track read type instructions indicated by the data hazard detection module as aliasing and track repeat aliasing of the tracked read type instructions, and to indicate displacements between instances of the tracked read type instructions and aliased instances of the write type instructions;

indicate write type instructions indicated by the data hazard detection module as aliasing; and

indicate instances of the write type instructions encountered in the rename unit that are indicated in the second structure.

59. (Previously Presented) The apparatus of claim 58 wherein the data hazard detection module includes a memory disambiguation buffer or a load/store.

60. (Previously Presented) The apparatus of claim 58 further comprising an instruction scheduling unit coupled with the rename unit and the data hazard detection module.

61. (Original) The apparatus of claim 58 wherein the structures include hardware tables and logical structures instantiable in memory.

62. (Previously Presented) An apparatus comprising:

an alias predictor, including one or more structures to host indications of write type instructions and particular instances of the write type instructions and read type instructions, respective execution displacements between particular instances of read and write type instructions, and register information of the particular write type instruction instances, the alias predictor operable to predict aliasings between read and write type instruction instances based, at least in part, on the structures and indications of detected aliasings between the instruction instances;

a rename unit coupled with the alias predictor, the rename unit to supply register information for write type instruction instances to the alias predictor; and

a data hazard detection unit coupled with the alias predictor, the data hazard detection unit to detect aliasing between particular instances of read and write type instructions and to indicate detected aliasings to the alias predictor.

63. (Original) The apparatus of claim 62, wherein the indications of particular instances of instructions include instruction instance addresses.

64. (Original) The apparatus of claim 63, wherein the instruction instances addresses include one or more of static identifiers and dynamic identifiers.

65. (Previously Presented) The apparatus of claim 62, wherein the structures comprise:

a first structure operable to indicate read type instructions with static identifiers thereof, respective execution displacements with potentially aliasing instances of write type instructions, and respective alias prediction confidence;

a second structure operable to indicate write type instructions with static identifiers thereof; and

a third structure operable to indicate particular instances of write type instructions with dynamic identifiers and register information thereof.

66. (Previously Presented) The apparatus of claim 65 further comprising:

the first and second structures operable to also indicate alias prediction validity; and the third structure operable to also indicate pending unverified alias predictions.